Carnegie Mellon
Software Engineering Institute

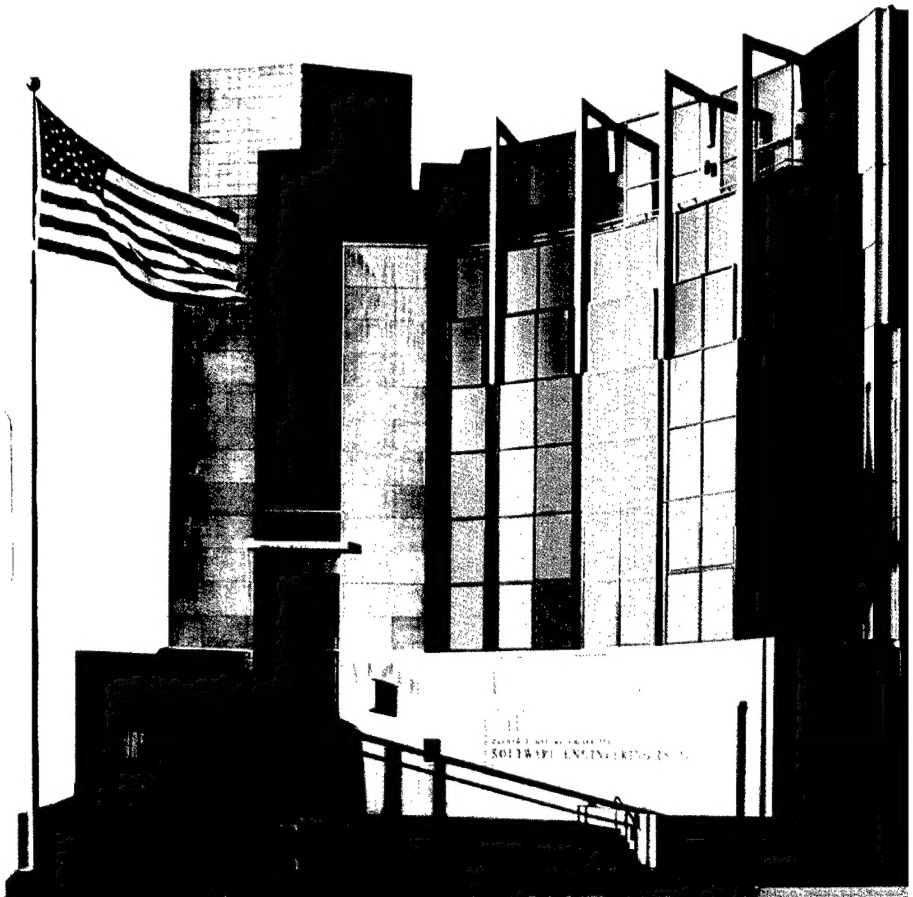
Requirements Engineering for Survivable Systems

Nancy R. Mead

September 2003

Networked Systems Survivability

DISTRIBUTION STATEMENT A
Approved for Public Release
Distribution Unlimited



Technical Note
CMU/SEI-2003-TN-013

Requirements Engineering for Survivable Systems

Nancy R. Mead

September 2003

Networked Systems Survivability

Technical Note
CMU/SEI-2003-TN-013

Contents

List of Figures

List of Tables

Abstract

This report describes the current state of requirements engineering for survivable systems, that is, systems that are able to complete their mission in a timely manner, even if significant portions are compromised by attack or accident. Requirements engineering is defined and requirements engineering activities are described. Survivability requirements are then explained, and requirements engineering methods that may be suitable for survivable systems are introduced. The report concludes with a summary and a plan for future research opportunities in survivable systems requirements engineering.

1 Background

In this report we will discuss the current state of requirements engineering for survivable systems. We start with some general definitions of requirements engineering and a discussion of requirements engineering activities. Then we introduce some requirements engineering concepts for survivable systems. We go on to discuss requirements engineering methods that may be suitable for survivable systems, both in the high assurance disciplines and in other areas as well. We conclude with a summary and plan for future research opportunities in survivable systems requirements engineering.

1.1 Definition of Requirements Engineering

Thayer and Dorfman [Thayer 97] define software requirements engineering as the science and discipline concerned with establishing and documenting software requirements. They state that it consists of software requirements elicitation, analysis, specification, verification, and management. They define software requirements *management* as "the planning and controlling of the requirements elicitation, specification, analysis, and verification activities." So, they consider requirements management to be part of requirements engineering.

In the Software Engineering Body of Knowledge (SWEBOK) [Sawyer 01], requirements engineering is described using a four-step process model, including requirements elicitation, analysis and negotiation, documentation, and validation. An output of this process is the set of agreed-upon requirements. Requirements elicitation is described as the first stage in building an understanding of the problem that the software is required to solve. Requirements analysis has to do with the process of analyzing requirements to detect and resolve conflicts among requirements, discover the bounds of the system and how it must interact with its environment, and elaborate user requirements to software requirements. Requirements negotiation has to do with resolving conflicts, such as those that might occur between stakeholders, or between requirements and resources. Validation is concerned with checks for omission, conflicting requirements, and ambiguities. This process is illustrated in Figure 1.

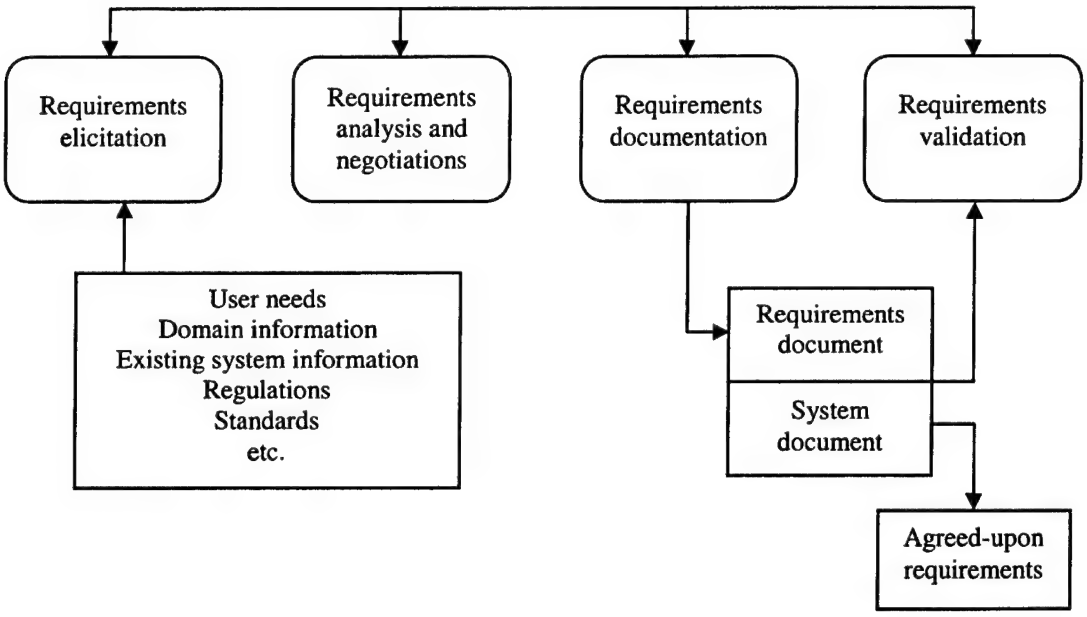


Figure 1: Coarse-Grain Requirements Engineering Process

1.2 Typical Requirements Engineering Activities

Davis describes the requirements (life-cycle) phase in terms of its activities [Davis 93]. The two major activities are problem analysis and product description. A "seed idea" initiates the problem-analysis activities, which include delineating constraints, refining constraints, tradeoffs between conflicting constraints, understanding the problem, and expanding information. This set of activities results in a relatively complete understanding of requirements, which initiates the product-description activities. During product description, consistency-checking and congealing take place, resulting in a consistent and complete software requirements specification. This is illustrated in Figure 2.

1.3 The Role of Requirements Management

Let's take a look at what is meant by requirements management. The Capability Maturity Model® for Software (SW-CMM®) provides good insight into the meaning of the term. In the SW-CMM, there are two goals in the requirements management key process area (KPA):

Goal 1: System requirements allocated to software are controlled to establish a baseline for software engineering and management use.

® Capability Maturity Model and CMM are registered in the U.S. Patent and Trademark Office by Carnegie Mellon University.

Goal 2: Software plans, products, and activities are kept consistent with the system requirements allocated to software [Paulk 94].

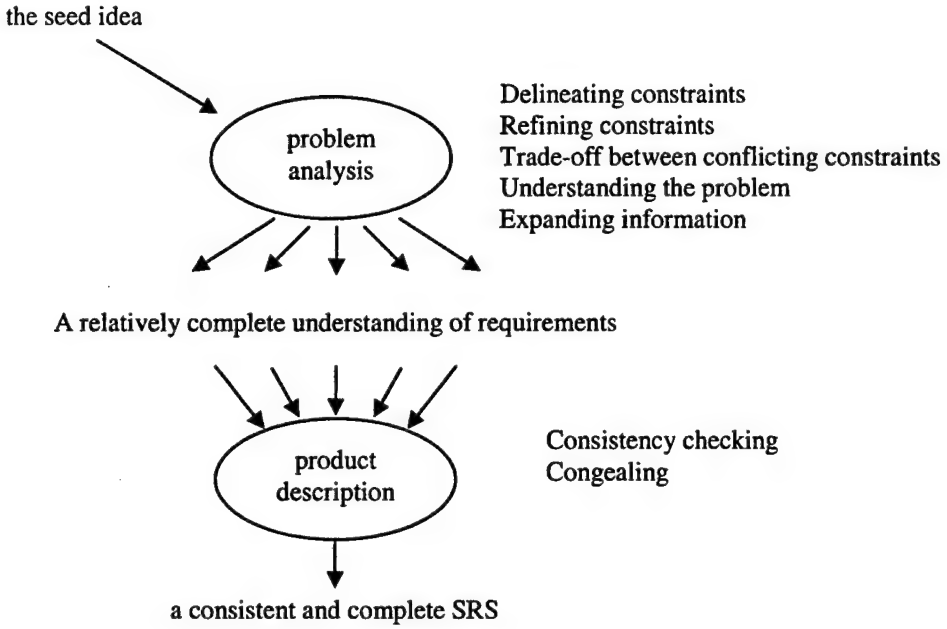


Figure 2: The Requirements Life-Cycle Activities

In the SW-CMM Version 2.0 Draft [SEI 97], the Requirements Management KPA was modified to include three goals:

Goal 1: Repeatable process (RM.GO.01). The activities for managing the allocated requirements are institutionalized to support a repeatable process.

Goal 2: Allocated requirements baseline (RM.GO.02). The software project's baseline of allocated requirements is established and maintained.

Goal 3: Allocated requirements consistency (RM.GO.03). The software project's plans, activities, and work products are kept consistent with the allocated requirements.

The more recent work on CMM Integration[SM] (CMMI®) models has expanded the focus on requirements engineering and requirements management.

It is pretty clear from both sets of goals that requirements management focuses on the life-cycle activities that must take place once the requirements have been established. The steps involved in establishing the requirements, on the other hand, fall more properly under the purview of requirements engineering.

[SM] CMM Integration is a service mark of Carnegie Mellon University.

® CMMI is registered in the U.S. Patent and Trademark Office by Carnegie Mellon University.

In the SWEBOK, requirements management is viewed as an activity that spans the whole life cycle. It involves change management and maintenance of the requirements in a state that accurately mirrors the software. The knowledge areas associated with requirements management are change management, requirements attributes, requirements tracing, and requirements documentation.

2 Requirements for Survivable Systems

Development of requirements for survivable systems allows us to build on existing knowledge. The recent series of RHAS [Mead 02, SEI 02] and SREIS [SREIS 02] workshops provides a focus on requirements for secure and survivable systems. In addition, there is an effort underway to recommend modifications to the Capability Maturity Model Integration (CMMI) models that are aimed at safety and security. In this section we present some definitional material on survivable systems in general, and more specifically on classes of survivable systems requirements. This section is extracted from a conference paper originally co-authored with Richard Linger and Howard Lipson [Linger 98].

2.1 Survivable Systems Definition

Survivability refers to the capability of a system to complete its mission in a timely manner, even if significant portions are compromised by attack or accident. In particular, survivability refers to the capability of a system to provide essential services in the presence of successful intrusion, and to recover compromised services in a timely manner after intrusion occurs. For example, a survivable financial network would maintain the integrity and availability of essential information, such as account and loan data, and services, such as transaction validation and processing. Integrity would be maintained even if particular nodes or communication links were incapacitated through intrusion or accident, and would recover compromised information and services in a timely manner. While survivability focuses on the preservation of mission capabilities, it includes issues of confidentiality and integrity as well. Because of the value of the CERT® intrusion knowledge base, this work has focused on attack and compromise by intelligent adversaries.

Experience with network systems has shown that no amount of hardening can guarantee invulnerability to attack. Despite best efforts, systems will continue to be breached. Thus it is vital to expand the current view of information systems security to encompass system behavior that contributes to survivability in spite of intrusions or accidents. Network systems must be robust in the presence of attack and able to survive attacks that cannot be completely repelled. The growing societal dependency on networks and the risks associated with their failure require that survivability be designed into these systems, beginning with effective survivability requirements analysis and definition.

In today's network environment, system security is largely dependent on the encryption of data and isolation through mechanisms such as firewalls. While the firewall approach is currently practical in a limited fashion, it will become increasingly inadequate to protect systems from intrusion in the rapidly expanding world of unbounded network computing.

Current systems are characterized by customer owned and controlled computing resources communicating over unbounded networks. In future systems, most computing resources will be resident within unbounded network infrastructures and will be controlled by a multitude of computing and communications service providers. These environments will be so unbounded as to render ineffective current security approaches, such as firewalls, that are based solely on isolation. In such environments, firewalls will be ineffective in detecting attacks, recovering from attacks, or helping systems survive intrusions and complete their missions in spite of malicious activity. Future unbounded systems will also embody dynamic architectures, capable of automated, real-time reconfiguration and adaptation in response to changing requirements and environments.

In summary, survivable network systems embody two essential characteristics. First, they preserve essential services under intrusion and recover full services in a timely manner. Second, they ensure survivability in environments characterized by unbounded networks and dynamic architectures. It is often the case that insufficient emphasis is placed on these survivability issues. As a result, the processes and techniques for addressing survivability are generally inadequate to deal with the threat. Concepts of system survivability provide a framework for integrating established disciplines of system reliability [Musa 87], safety [Leveson 95], security [Clark 93], and fault tolerance [Mendiratta 96], as well as emerging disciplines such as dynamic system adaptation, diversification,[1] and trust maintenance.

2.2 Survivability Requirements

Figure 3 depicts an iterative model for defining survivable system requirements. We recognize that survivability must address not only requirements for software functionality, but also requirements for software usage, development, operation, and evolution. Thus, five specific types of requirements definition are relevant to survivable systems in the model of Figure 3, as discussed below.

[1] From "Systematic Generation of Stochastic Diversity in Survivable System Software," by R.C. Linger, currently submitted for publication.

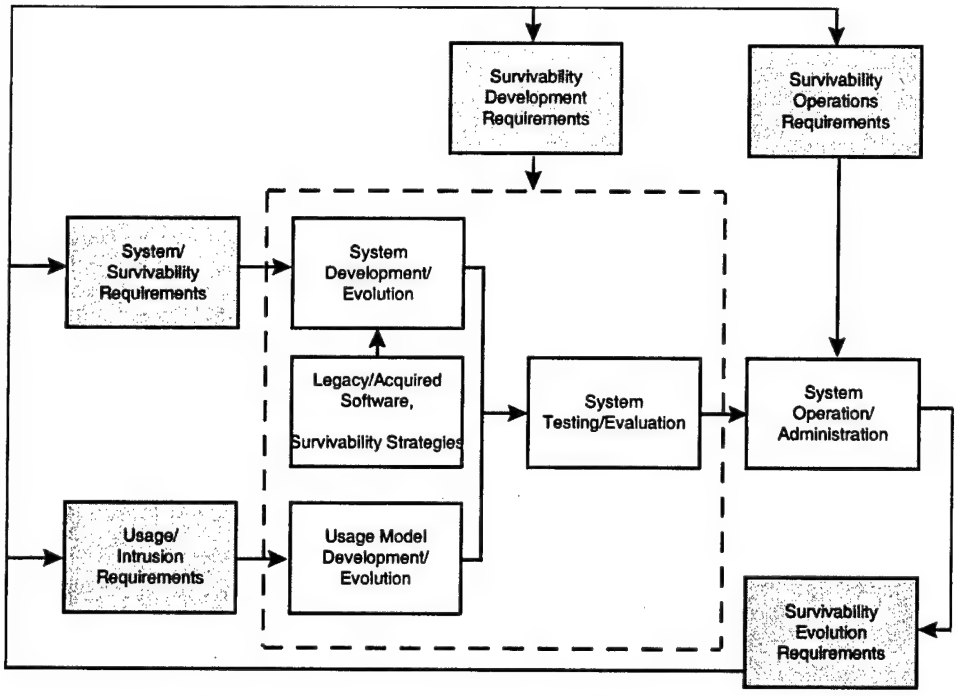


Figure 3: Requirements Definition for Survivable Systems

2.2.1 System/Survivability Requirements

In this exposition, the term *system requirements* refers to traditional user functions that a system must provide. For example, a network management system must provide user functions for monitoring network operations, adjusting performance parameters, and so forth. System requirements also include non-functional aspects, such as timing, performance, and reliability. The term *survivability requirements* refers to system capabilities for the delivery of essential services in the presence of attacks and intrusions, and recovery of full services.

Figure 4 depicts the integration of survivability requirements with system requirements at node and network levels. First, survivability requires that system requirements be organized into essential services and non-essential services, perhaps organized in terms of user categories or business criticality. Essential services must be maintained even during successful intrusions; non-essential services are to be recovered after intrusions have been dealt with. Essential services may be further stratified into a number of levels, each embodying fewer and more vital services, as a function of increasing severity and duration of intrusion. It is also possible that the set of essential services may vary in a more dynamic manner, depending on a particular attack scenario and the resulting situation. In this dynamic case, services that are essential under one scenario may not be essential under another, resulting in different combinations of essential services that are scenario dependent. Thus, definitions of requirements for essential services must be augmented with appropriate survivability requirements. As shown in Figure 3, survivable systems may also include legacy and commercial off-the-shelf (COTS) components not originally developed with

survivability as an explicit objective. Such components may provide both essential and non-essential services and may engender special functional requirements for isolation and control through wrappers and filters to help permit safe use in a survivable system environment.

Second, Figure 4 shows that survivability itself imposes new types of requirements on systems for *resistance* to, *recognition* of, and in particular, *recovery* from intrusions and compromises [Ellison 97]. These survivability requirements are supported by a variety of existing and emerging *survivability strategies*, as noted in Figure 1 and discussed in more detail below. Finally, Figure 2 depicts *emergent behavior requirements* at the network level. These requirements are characterized as "emergent" because they result from the collective behavior of node services communicating across the network, without benefit of centralized control or information. These requirements deal with the survivability of overall network capabilities, such as capabilities to route messages between critical sets of nodes regardless of how intrusions may damage or compromise network topology.

We envision survivable systems as being capable of adapting their behavior, function, and resource allocation in response to intrusions. When necessary, for example, functions and resources devoted to non-essential services could be reallocated to the delivery of essential services and intrusion resistance, recognition, and recovery. Requirements for such systems must specify the behavior for adaptation and reconfiguration in response to intrusions.

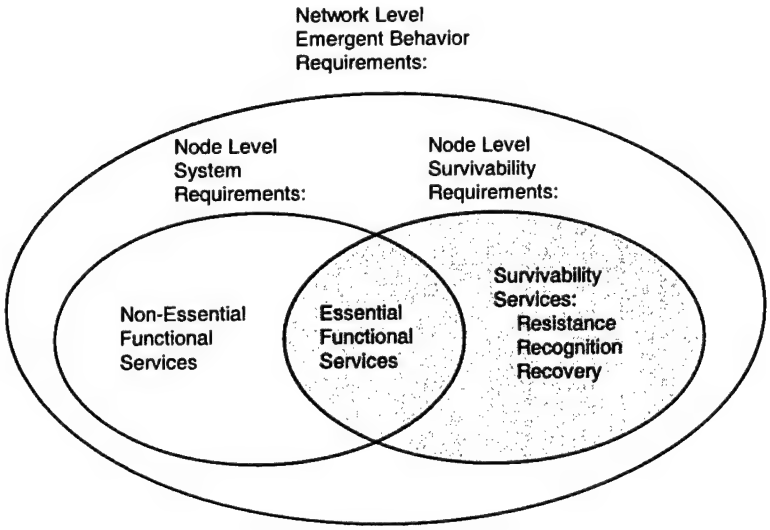


Figure 4: Integrating Survivability Requirements with System Requirements

Systems can exhibit large variations in survivability requirements. Small local networks may have few or even no essential services, with acceptable manual recovery times measured in hours. Large-scale networks of networks may be required to maintain a core set of essential services, with automated intrusion detection and recovery times measured in minutes. Embedded command and control systems may require essential services to be maintained in real time, with recovery periods measured in milliseconds. The attainment and maintenance

of survivability consumes resources in system development, operation, and evolution. Survivability requirements for a system should be based on the costs and risks to an organization associated with loss of essential services.

2.2.2 Usage/Intrusion Requirements

Survivable system testing must demonstrate the performance of essential and non-essential system services, as well as the survivability of essential services under intrusion. Because system performance in testing (and operation) depends totally on the usage to which it is subjected, an effective approach to survivable system testing is based on usage scenarios derived from usage models [Mills 92, Trammell 95].

Usage models are developed from *usage requirements*, which specify legitimate usage environments and all possible usage scenarios. Usage requirements for essential and non-essential services must be defined in parallel with system and survivability requirements. Furthermore, intrusion usage must be treated on a par with legitimate usage, and intrusion *requirements*, which specify intrusion usage environments and all possible scenarios of intrusion use, must be defined as well. In this approach, intrusion usage is modeled in conjunction with the legitimate use of system services. Figure 5 depicts the relationship between legitimate and intrusion usage. Intruders may engage in usage scenarios beyond legitimate scenarios, but may also employ legitimate usage for purposes of intrusion if they become privileged to do so.

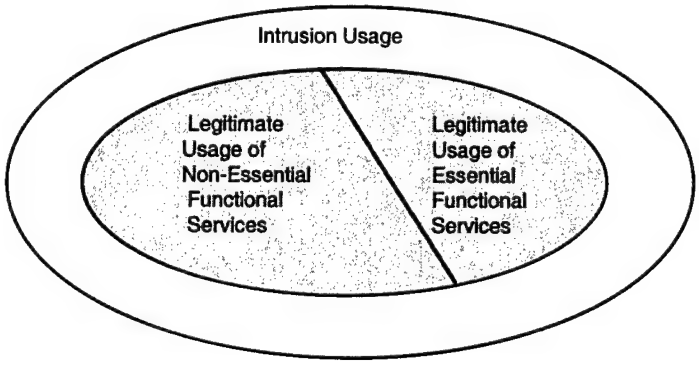


Figure 5: The Relationship Between Legitimate and Intrusion Usage

2.2.3 Development Requirements

Survivability places stringent requirements on system development and testing practices. Software errors can have a devastating effect on system survivability and provide ready opportunities for intruder exploitation. Sound engineering practices are required to create survivable software. We assert the following five principles, four technical and one organizational, as example requirements for survivable system development and testing practices:

- precise specification of required functions in all possible circumstances of use
- correctness verification of implementations with respect to function specifications
- specification of function usage in all possible circumstances of use, including intruder usage
- testing and certification based on function usage and statistical methods
- establishment of permanent readiness teams for system monitoring, adaptation, and evolution

Sound engineering practices are required to deal with legacy and COTS software components as well.

2.2.4 Operations Requirements

Survivability places demands on the requirements for system operation and administration to define and administer survivability policies, monitor system usage, respond to intrusions, and evolve system functions as necessary to ensure survivability as usage environments and intrusion patterns change over time.

2.2.5 Evolution Requirements

System evolution is an inevitable necessity to respond to user requirements for new functions and increasing intruder knowledge of system behavior and structure. In particular, survivability requires that system capabilities evolve more rapidly than intruder knowledge to prevent the accumulation of information about invariant system behavior and structure needed to achieve successful penetration and exploitation.

2.3 Requirements Definition for Essential Services

The preceding discussion distinguishes between essential and non-essential services. At the highest level, each system requirement needs to be examined to determine whether it corresponds to an essential service. The set of essential services must form a viable subsystem relative to the original system. In the event that levels of essential services are required, the set of services provided at each level must be examined for completeness and coherence. As noted above, the set of essential services could vary in a more dynamic way, depending on particular scenarios or situations. In addition, requirements must be defined for transitioning to and from essential service modes.

In distinguishing essential and non-essential services, all the usual requirements definition processes and methods can be applied. Elicitation techniques such as those described in *Software Requirements Engineering* [Thayer 97] can help to identify essential services and a tradeoff and cost/benefit analysis can help to determine appropriate sets of services that

sufficiently address business survivability risks and vulnerabilities. Provisions for the traceability of survivability requirements through design and code must be established, and special test cases would be required as well. As noted above, the simulation of intrusion through intruder usage scenarios would be included in the testing strategy.

2.4 Requirements Definition for Survivability Services

Penetration, exploration, and exploitation create a spiral of increasing intruder authority and an ever-widening circle of compromise. For example, penetration at the user level is typically employed as a means to explore for root-level vulnerabilities. User-level authorization is then employed to exploit those vulnerabilities to achieve root-level penetration. Furthermore, a compromise of the weakest host in a networked system allows that host to be used as a stepping-stone to comprise other more protected hosts.

Requirements definitions for resistance, recognition, and recovery services embody selected survivability strategies to deal with these phases of intrusion. Some strategies, such as firewalls, are the product of extensive research and development and are used extensively in current bounded networks. The following new strategies are emerging as necessary responses to the unique challenges of unbounded networks.

2.4.1 Resistance Service Requirements

Resistance refers to the capability of a system to deter attacks. Thus, resistance is important in the penetration and exploration phases of an attack, prior to the point where actual exploitation occurs. Current strategies for resistance include the use of firewalls, authentication, and encryption. Diversification is an example of a strategy that will likely become important in future unbounded networks.

Diversification requirements must define a planned variation in survivable system function, structure, and organization, together with a means for achieving it. Diversification is intended to create a "moving target" to intruders and to render ineffective the accumulation of system knowledge as an intrusion strategy. Diversification also eliminates intrusion opportunities associated with multiple nodes that execute identical software and thus exhibit identical vulnerabilities. Such systems offer tempting economies of scale to intruders, since all nodes can be penetrated once one node has. Diversification requirements can include a variation in programs, retained data, and network routing and communication. For example, systematic means can be defined to randomize software programs while preserving functionality.[2]

[2] From "Systematic Generation of Stochastic Diversity in Survivable System Software" by R. C. Linger, currently submitted for publication.

2.4.2 Recognition Service Requirements

Recognition refers to the capability to recognize attacks or to recognize the probing that may precede attacks. The ability to react or adapt in the face of intrusion is central to the capacity of a system to survive an attack that cannot be completely repelled. Reaction or adaptation is impossible without some form of recognition, and thus recognition is essential in all three phases of attack.

A substantial body of research and development exists in this area. Current strategies for attack recognition include not only state-of-the-art work in intrusion detection, but also more mundane but nevertheless effective techniques of logging and frequent auditing, as well as follow-up investigations of reports generated by ordinary error-detection mechanisms. There are two types of advanced intrusion-detection techniques: anomaly detection and pattern recognition. Anomaly detection is based on models of normal user behavior. These models are often established through the statistical analysis of usage patterns. Deviations from normal usage patterns are flagged as suspicious. Pattern recognition is based on models of intruder behavior. User activity that matches a known pattern of intruder behavior raises an alarm.

The requirements for future survivable networks will likely employ additional strategies, such as self-awareness, trust maintenance, and black-box reporting. Self-awareness refers to the establishment of a high-level semantic model of the computations that a component or system is executing or has been asked to execute. A system or component that "understands" what it is being asked to do is in a position to refuse those actions that would be dangerous, compromise a security policy, or adversely impact the delivery of minimum essential services. By trust maintenance, we refer to a requirement for periodic queries among the components of a system (e.g., among the nodes in a network) to continually test and validate trust relationships. The detection of intrusion signs would trigger an immediate test of trust relationships. Black-box reporting refers to a dump of system information that could be retrieved from a crashed system or component for analysis by the rest of the system to determine the cause of the crash (e.g., design error or specific intrusion type), and thereby prevent other components from suffering the same fate.

In summary, a survivable system design must include explicit requirements for attack recognition. These requirements will ensure the use of one or more of the strategies described above, through the specification of architectural features, automated tools, and manual processes. Since intruder techniques are constantly advancing, it is essential that recognition requirements be subject to frequent review and continuous improvement.

2.4.3 Recovery Service Requirements

Recovery refers to a system's ability to restore services after an intrusion has occurred and to improve its capability to resist or recognize future intrusion attempts. Recovery also contributes to a system's ability to maintain essential services during intrusion.

The requirements for recoverability are what most clearly distinguish survivable systems from merely secure systems. Traditional computer security leads to the design of systems that rely almost entirely on hardening (i.e., resistance) for the protection of system resources and services. Once security is breached, damage may soon follow with little to stand in the way. As stated earlier, the ability of a survivable system to react or adapt in the face of an active intrusion is central to the capacity of a system to survive an attack that cannot be completely repelled. Thus, recovery is crucial during the exploration and exploitation phases of intrusion.

Recovery strategies in use today include the replication of critical information and services, the use of fault-tolerant designs, and a variety of backup systems for hardware and software, including maintaining master copies of critical software in isolation from the network. Future recovery strategies will most certainly include dynamic system adaptation, which will not only help a system recover from a current attack, but also permanently improve a system's ability to resist, recognize, and recover from future intrusion attempts. For example, a recoverability requirement for a survivable system may include infrastructure support for the capacity to inoculate the entire system against newly discovered security vulnerabilities, through the automated distribution and application of security fixes to all network elements. Similarly, recoverability requirements may specify that intrusion-detection rule sets are to be updated in a timely manner, in response to reports of known intruder activity from an authoritative source of security information, such as the CERT Coordination Center.

In summary, explicit requirements for recovery are crucial for the design of a survivable system. Recovery requirements make adaptability an integral part of a system's design. As was the case for resistance and recognition requirements, the constant evolution of intruder techniques makes it essential that recovery requirements be subject to frequent review and continuous improvement.

2.5 Summary

In this section we have discussed some definitional work towards identifying and classifying survivable systems requirements. We have also identified strategies that can assist in identifying survivable systems requirements, and which ultimately result in systems that are more survivable. There have been other general approaches to requirements engineering for security requirements that are also worth reading about [Firesmith 03b]. In addition, our life-cycle research emphasizes the importance of survivability requirements engineering [Mead 01].

3 Methods and Practices that Support Requirements Engineering for Survivable Systems

There has been a significant amount of work on methods to support requirements for survivable systems. In this section, we sketch out some of these methods. This will allow us to build on existing work, and to select promising methods for experimentation.

3.1 Some existing methods and practices

3.1.1 Misuse and Abuse Cases

A security "misuse" case [Alexander 03, Sindre 00, Sindre 02] a variation on a use case, is used to describe a scenario from the point of view of the attacker. Since use cases have proven useful in documenting normal use scenarios, they can also be used to document intruder usage scenarios, and ultimately used to identify security requirements or security use cases [Firesmith 03a]. A similar concept has been described as an "abuse" case [McDermott 01, McDermott 99].

One obvious application of a misuse case is in eliciting requirements. Since use cases are used successfully for eliciting requirements, it follows that misuse cases can be used to identify potential threats and to elicit security requirements. In this application, the traditional user interaction with the system is diagrammed simultaneously with the hostile user's interactions. An example of this is shown in Figure 6 [Alexander 03].

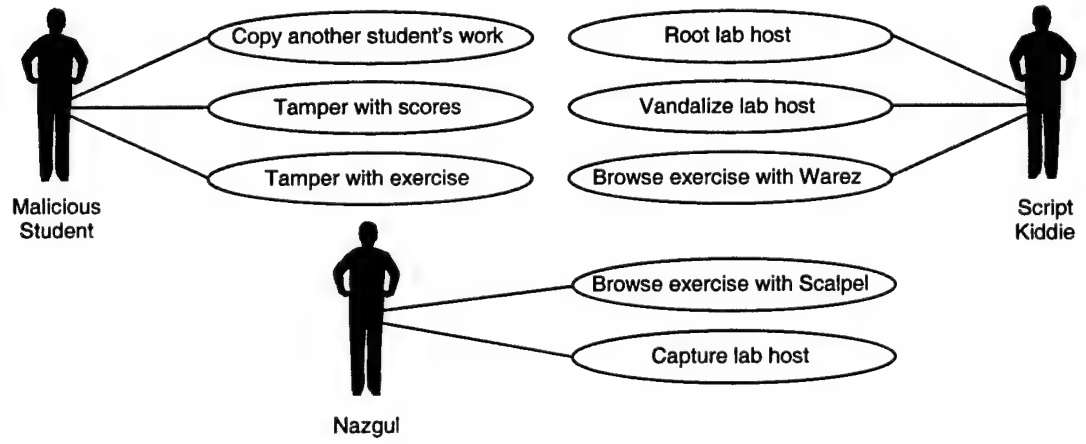


Figure 6: Abuse Case Diagram for an Internet-Based Information Security Laboratory

Alternatively, abuse cases tend to show the "abuse" side of the system, in contrast to traditional use cases. The contrast between use and abuse cases is shown in Table 1 [McDermott 99].

Table 1: Contrast between Use and Abuse Cases

Use Case	Abuse Case
• A complete transaction between one or more actors and a system • UML-based use case diagrams • Typically described using natural language	• A family of complete transactions between one or more actors and a system that results in harm • UML-based use case diagrams • Typically described using natural language. A tree/DAG diagram may also be used. • Potentially one family member for each kind of privilege abuse and for each component that might be exploited • Includes a description of the range of security privileges that may be abused • Includes a description of the harm that results from an abuse case

Using these concepts, Firesmith develops tabular examples of security use cases. His own version of the differences between security use cases and misuse cases is shown in Table 2. A complete example is shown in Table 3 [Firesmith 03a].

Table 2: Differences Between Misuse Cases and Security Use Cases

	Misuse Cases	Security Use Cases
Usage	Analyze and specify security threats	Analyze and specify security requirements
Success Criteria	Misuser succeeds	Application succeeds
Produced By	Security team	Security team
Used By	Security team	Requirements team
External Actors	Misuser, user	User
Driven By	Asset vulnerability analysis Threat analysis	Misuse cases

3.1.2 Formal Methods

Formal methods are typically used in specification and verification of secure and survivable systems. From a life-cycle viewpoint, the specification typically represents either formal requirements or a formal step between informal requirements and design.

Some of the methods are applied to security standards, such as the Common Criteria and IP Security Protocol (IPSec). Organizational objectives are translated into the specification of all relevant security functions in a planned system. The subset of specifications to be implemented is identified and further assessment or risk analysis takes place [Leiwo 99a].

Table 3: The Access Control Use Case

Use Case: Access Control		
Use Case Path: Attempted Spoofing Using Valid User Identity		
Security Threat: The system authenticates and authorizes the misuser as if the misuser were a valid user.		
Preconditions: 1) The misuser has a **valid** means of user identification. 2) The misuser has an **invalid** means of user identification.		
Misuser Interactions	**System Requirements**	
	System Interactions	**System Actions**
	The system shall request the misuser's means of identification and authentication.	
The misuser provides a valid means of user identity but an invalid means of user authentication.		
		1) The system shall misidentify the misuser as a valid user. 2) The system shall **not** authenticate and authorize the misuser.
	The system shall reject the misuser by canceling the transaction.	
Postconditions: 1) The system shall not have allowed the misuser to steal the user's means of authentication. 2) The system shall not have authenticated the misuser as a valid user. 3) The system shall not have authorized the misuser to perform any transaction that requires authentication. 4) The system shall have recorded the access control failure.		

The Common Criteria are used during the second or evaluation phase. The Kruger-Eloff process, based on the Common Criteria, is used for evaluation of information security. Another effort [Fu 01] contributes to correctness and conflict resolution of IPSec security policy. This method allows definition of a high-level security requirement that can be used to detect conflicts among IPSec policies, and also aids in automation of the policy specification process for IPSec policies. Another method focuses more generally on information security policy specification [Ortalo 98]. A formal specification language is described, and in a case study the method is applied to the description of security requirements for a medium-size banking organization. This method provides flexibility and expression so as to correspond to specific organizational needs.

One study focuses on security policies based on known potential secrets [Biskup 01]. In this study, security requirements are explicitly defined and formally made comparable with requirements for policies based on secrecy. An evaluation strategy based on lying is adapted to the framework and formally proven to meet the security requirements. Weak conditions

for the functional equivalence of lying and refusal are identified, with respect to the information learned from answers to queries, along with the user's assumed initial knowledge. As an example for the dynamic approach based on lies, the authors study whether users can determine which query answers are reliable. A variant of the refusal-based approach is analyzed and compared with the lying approach.

The B formal method is used specifically to support the design and validation of the transaction mechanism for smart card applications. The mathematical proofs provide confidence that the design of the transaction mechanism satisfies the security requirements [Sabatier 99].

An interesting contribution is a model that focuses on modeling the organization in which information security is developed [Leiwo 99b]. The organization is described in layers of abstraction. In addition, a notation for expressing security requirements is described, under a framework of harmonization functions and merging of requirements. A case study that focuses on the security requirements for sharing of patient data among hospitals and medical practitioners is described.

3.1.3 Use of Trees for Modeling and Analysis

Several approaches depend on the use of trees for modeling survivability requirements. Attack trees can be used in requirements elicitation [Ellison 03, Moore 01] and fault trees have been used in requirements analysis [Helmer 02, Kienzle 98].

The notion of attack trees as a method for modeling attacks has been described extensively in the literature [Schneier 00]. The work by Ellison and Moore [Ellison 03, Moore 01] explores the use of attack trees in development of intrusion scenarios, which can then be used to identify requirements. A small attack tree example is shown in Figure 7.

Although aimed initially at architectural analysis, it is easy to see how attack trees can also be used to help answer the survivability questions:

- How can we detect an attacker during an attempted attack or after a successful attack?
- How can we recover from any compromise?
- How can we adapt the system so that the intrusion cannot happen again?

Fault trees use a set of special symbols to depict intrusions. The symbols used in the paper by Helmer et al. are illustrated in Figure 8. Fault trees are used for modeling intrusions and intrusion steps, such as penetration using buffer overflow, illustrated in Figure 9.

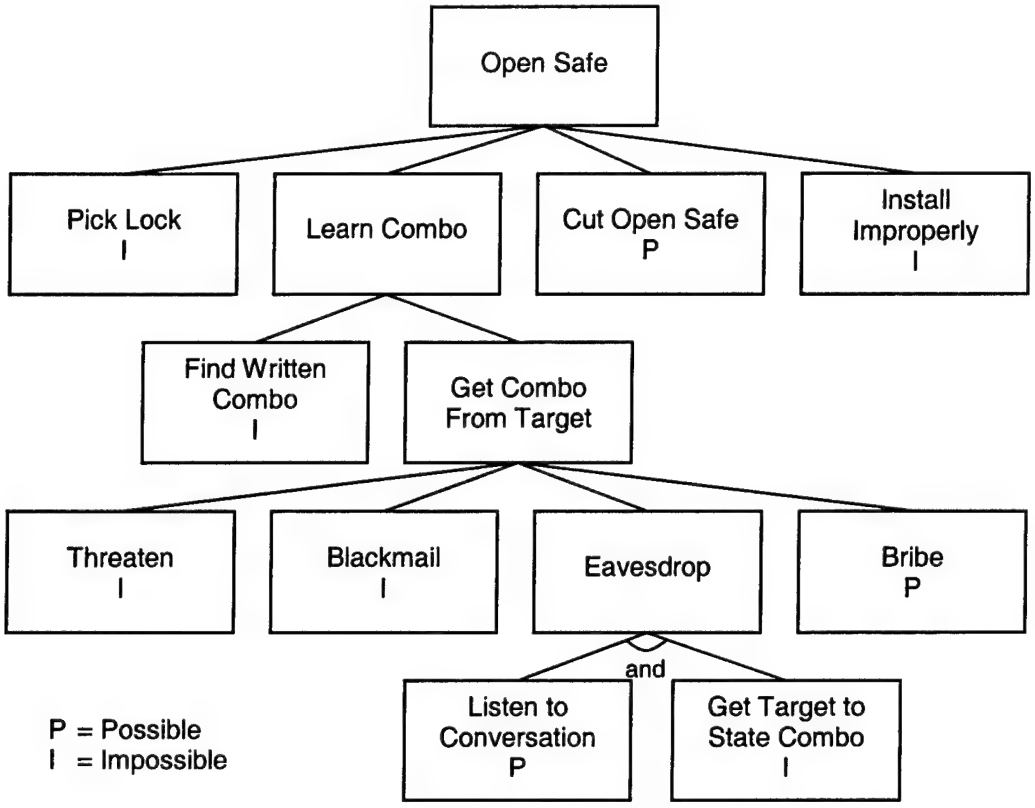


Figure 7: Attack Tree Example

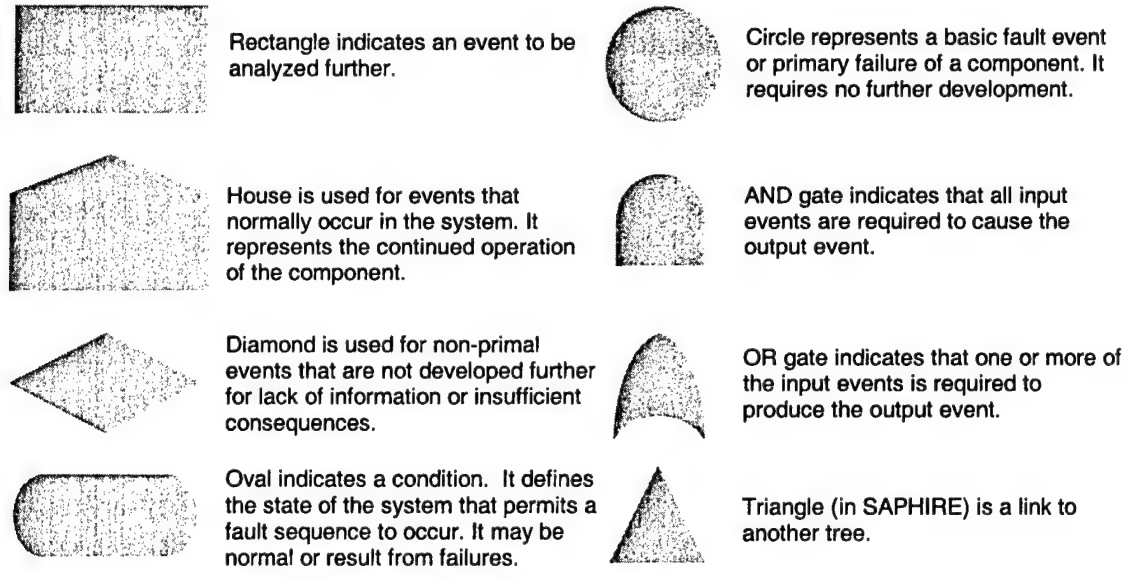


Figure 8: Relevant Fault Tree Symbols

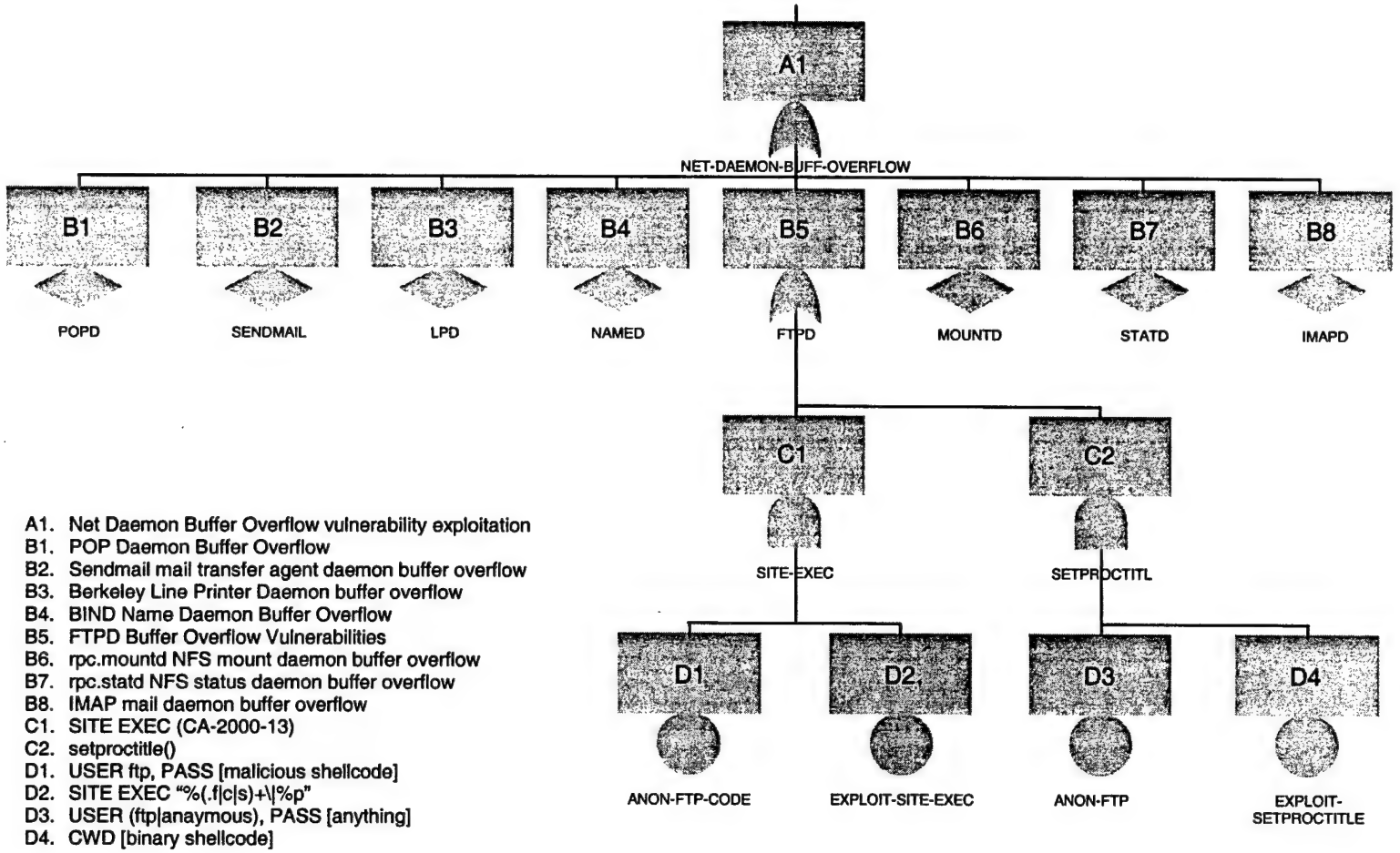


Figure 9: Penetration Fault Tree: Using Buffer Overflow in Network Daemons

Once fault trees have been used to model intrusions, they can also be used to help identify requirements for intrusion detection systems, as described in the paper. Alternatively, fault tree analysis can be used to identify other security and survivability requirements, once the fault trees have been used to model intrusion behavior. Formal use of fault trees suggests the possibility of formal analysis, which could be a great advantage in developing a set of consistent and complete requirements.

The Methodically Organized Argument Tree (MOAT) methodology [Kienzle 98] has integrated existing techniques into a risk-driven process model. An argument tree incorporates the desired property, formal proofs, informal reasoning, assumptions, axioms, lemmas, and component proofs, thus providing a framework for analysis. Tree construction follows a sequence of steps that incorporates the following processes: Initialization, Justification, Order of Analysis, Decomposition into Subgoals, Decomposition into Alternatives, Refinement, Backtracking, Termination Criteria, and Assessment.

3.1.4 Software Cost Reduction

Software Cost Reduction (SCR) is a formal method based on a tabular representation of specifications, and analysis of the requirements for complex systems. It was originally developed to document the behavior of the A-7E aircraft [Heninger 78, Heninger 80], and has been augmented with a tool suite and applied to many complex and safety-critical systems [Bharadwaj 03, Heitmeyer 96, Heitmeyer 00]. Figure 10 shows the relationship between the System Requirements Specification (SRS), the System Design Specification (SDS), and the Software Requirements Specification (SoRS).

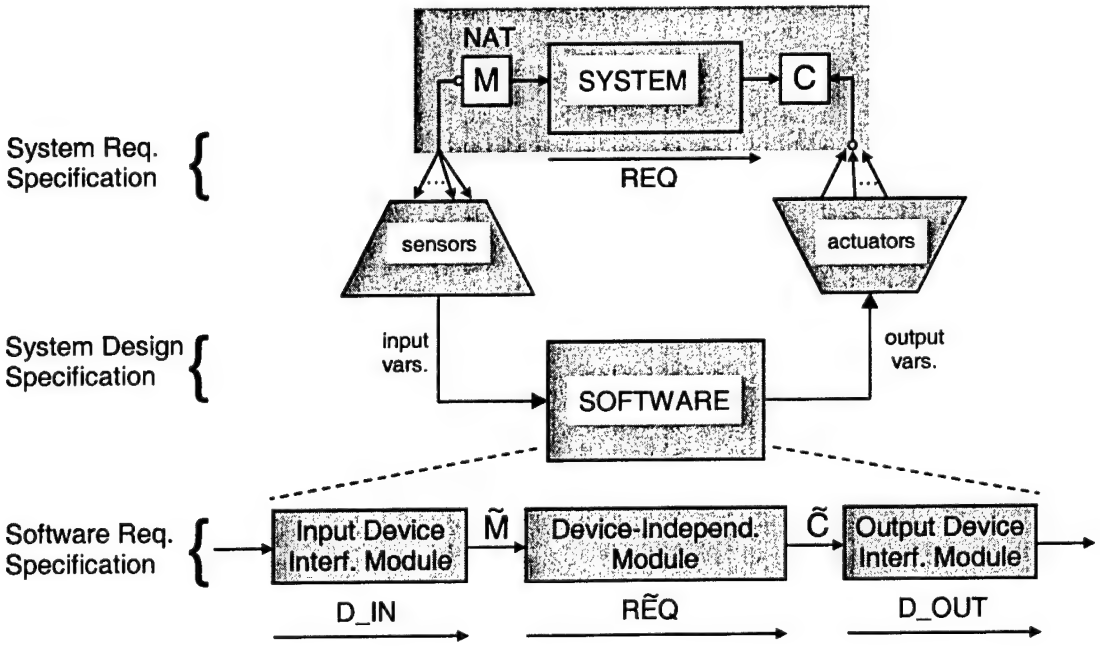


Figure 10: Relationship Between the SRS, the SDS, and the SoRS

This decomposition is commonly used in many large DoD and other government systems. The SCR notation is used for specification. According to Heitmeyer and Bharadwaj,

> "To specify the required system behavior in a practical and efficient manner, the SCR method uses terms and mode classes. A *term* is an auxiliary variable that helps keep the specification concise. A *mode class* is a special case of a term, whose values are modes. Each mode defines an equivalence class of system states, useful in specifying the required system behavior. In SCR specifications, we often use prefixes in variable name. In SCR specifications, we often use the following prefixes in variable names: "m" to indicate monitored variables, "t" for terms, "mc" for mode classes, "c" for controlled variables, "i" for input variables and "o" for output variables.
>
> "Conditions and events are important constructs in SCR specifications. A *condition* is a predicate defined on one or more state variables (a *state variable* is a monitored or controlled variable, a mode class, or a term). An *event* occurs when a state variable changes value."

Table 4 is an example of an SCR table.

Table 4: Condition Table Defining the Value of Term tRemLL

Mode Class = mcStatus			Trac.
Mode	**Condition**		
unoccupied	true	false	FM3
occupied	mIndoorLL > tCurrentLSVal	mIndoorLL ≤ tCurrentLSVal	FM1
temp_empty	mIndoorLL > tCurrentLSVal OR tOverride	mIndoorLL ≤ tCurrentLSVal AND NOT tOverride	FM1, FM6
tRemLL	0	tCurrentLSVal – mIndoorLL	FM1

For survivable systems that require a rigorous specification method, SCR would seem to be a good choice. It is probably not as useful in the early requirements stages, for example during elicitation, and may have the most utility in the specification activity that tends to occur between requirements and design activities.

3.1.5 Requirements Reuse

The promise of requirements reuse is attractive in the information security area. Many organizations don't really know how to get started in identifying and specifying security requirements, so the idea of a library of reusable security requirements is very appealing. An initial approach has been described [Toval 02] and follow-on work is in progress. The approach describes a possible scheme for a reuse repository and case study examples.

3.1.6 Risk Analysis

One of the challenges in survivable systems has been development of quantitative methods of assessing risk. Since many organizations are unaware of attacks on their systems, how can they quantify the risk associated with them? When there is awareness of attacks, it is typically the virus or worm attacks, scanning, denial-of-service, or other attacks that are easy to measure using tools. Sophisticated attacks are seldom detected, so how can their risk be quantified? Without quantifiable risk analysis, how can requirements be developed in a sensible way to address those risks?

A number of risk analysis methods are currently in use or development. The OCTAVE method [Alberts 03] provides a framework for survivability risk analysis, but is fairly general when it comes to requirements. Recent work on multi-attribute risk assessment [Butler 02] and on a risk-centric decision process [Feather 03] provides some promise in addressing the risk analysis problem.

OCTAVE risk evaluation has three phases and eight processes. The phases are

- Phase 1: Build Asset-Based Threat Profiles
- Phase 2: Identify Infrastructure Vulnerabilities
- Phase 3: Develop Security Strategy and Plans

The eight processes are

- Elicitation workshop for senior managers
- Elicitation workshop for operational managers
- Elicitation workshop for general staff, information technology staff
- Creating threat profiles
- Identifying key components
- Evaluating selected components
- Conducting the risk analysis
- Developing a protection strategy

The OCTAVE process provides a very thorough risk analysis of existing systems. By and large, the organization provides the resources for this process, with training and assistance from external facilitators. It is a major investment for the organization and provides for ongoing risk analysis. Requirements are not a major focus of the method, which is geared towards large operational systems.

In multi-attribute risk assessment, a security manager's experience is used to estimate and then prioritize security risks and the associated security requirements. Case studies suggest

that security managers do a credible job of assessing existing risks, based on data associated with actual and predicted attacks. This data is then used to help to quantify the possible negative outcomes, such as lost productivity or public reputation, and using a weighting scheme, priorities are associated to these risks. An example of these outcome attributes [Butler 02] is shown in Table 5.

Table 5: Outcome Attributes

Outcome Attribute	Rank	Assessed Preference	Weight
Lost Productivity	1	100	.42
Public Reputation	2	80	.33
Regulatory Penalties	3	40	.17
Lost Revenue	4	20	.08

The risks correspond to threats, which in turn drive risk mitigation strategies that are embodied in security requirements. The method requires a relatively small level of effort on the part of the assessor, and has been used in case studies with several organizations. Security managers provide data and participate in interviews as part of the assessment activity.

In risk-centric decision processes[3] [Feather 03], a three-day workshop is conducted to identify risks and their mitigation strategies and to decide which mitigation strategies to pursue. On the first day, objectives, risks, and impacts are identified. On the second day, mitigation strategies and the corresponding reductions in risk are identified. On the third day, decision-making is made on which mitigations to perform, which objectives to discard, and the resources needed to support the strategies. Getting the right set of participants to identify all of these elements is key, and the ability to come to a decision in a relatively short time is a significant benefit. A tool (DDP) is provided to support the decision process (see http://ddptool.jpl.nasa.gov).

Another approach suggests developing requirements based on two dimensions: "determining information security concern percentages" and "the impact of events and the impact on services, products and processes" [Gerber 01]. The concerns are confidentiality, integrity, availability, auditability, and authenticity. The impacts resulting from a security incident are considered in the second dimension. Examples are given. In this approach, it is suggested

[3] Feather, M. S. & Cornford, S. L. "Quantitative Risk-Based Requirements Reasoning." To appear in *Requirements Engineering* (Springer-Verlag) in 2003.

that risk analysis is no longer adequate to determine the required level of information security.

3.1.7 Examples of Security Requirements

There are a number of papers in the literature that have provided examples of security requirements. Some of these have been discussed in the previous sections. However, there are others that don't quite fit the earlier discussion, but are nevertheless noteworthy. We include some of these here. These papers tend to provide security requirements examples for specific domains. These include the domains of ATM network security [Leitold 99], electronic commerce security [Labuschagne 00], security requirements for e-business processes [Knorr 01], security requirements for management systems using mobile agents [Reiser 00], mediation [Biskup 99], and support for multi-level secure and real-time databases [Son 98].

3.2 Selection of Promising Methods and Practices for Security and Survivability Requirements Engineering

In our work with acquirers of systems and with practitioners, we find that many organizations do not have a good awareness of security and survivability requirements. They do not identify them during requirements development, so the question of analysis, specification, and verification is a moot point. We therefore feel that elicitation is a good place to start our quest for survivable requirements engineering. Misuse/abuse cases, security use cases, and attack trees show some promise for this purpose. In addition, techniques such as structured interviews, focus groups, and prioritization techniques will play a role.

The earlier work in partitioning survivable requirements into classes of requirements should prove useful in breaking down the problem. We also believe that formal specification methods such as model checking and SCR will play a role, leading to formal design methods such as flow-service-quality (FSQ) [Linger 02, Linger 03] and the associated function extraction methodology (FX).

4 Summary and Plans

Although we have discussed many potentially useful techniques, our discussion is not exhaustive. We hope that future reports will document additional interesting and useful techniques that are already available. In addition, much research remains to be done in this area. While many of the methods seem promising for survivable system requirements, an integrated methodology that covers all survivable system requirements needs (from elicitation to analysis, specification, and validation, incorporating requirements management) does not exist at present. Many of the methods that are used in different requirements activities need to be tested on survivable systems problems as well. Of course, in any survivable system, there is the question of scale. Large, unbounded systems need methodological and tool support that goes beyond small research artifacts.

Our plan is to test selected methodologies as a proof-of-concept on survivable systems projects, and to refine promising methods as a result. An adjunct activity is to use these building blocks to develop an end-to-end process for survivable system requirements engineering. Since many operational systems problems are traceable to requirements problems, we hope to enable development of systems that are more survivable by successfully using requirements engineering methods in their development.

References

[Alberts 03] Alberts, C. & Dorofee, A. *Managing Information Security Risks: The OCTAVE Approach.* New York: Addison Wesley, 2003.

[Alexander 03] Alexander, I. "Misuse Cases: Use Cases with Hostile Intent." *IEEE Software 20*, 1 (January-February 2003): 58-66.

[Bharadwaj 03] Bharadwaj, R. "How to Fake a Rational Design Process Using the SCR Method," 3-4. *SEHAS'03 International Workshop on Software Engineering for High Assurance Systems*. Portland, OR, May 9-10, 2003. Pittsburgh, PA: Carnegie Mellon University, Software Engineering Institute, 2003. <http://www.sei.cmu.edu/community/sehas-workshop/bharadwaj/>.

[Biskup 99] Biskup, J.; Flegel, U.; & Karabulut, Y. "Secure Mediations: Requirements and Design," 127-140. *Database Security XII: Status and Prospects*. Edited by S. Jajodia. Twelfth International Working Conference on Database Security, Chalkidiki, Greece, July 15-17, 1998. Norwell, MA: Kluwer Academic Publishers, 1999 (ISBN 0792384881).

[Biskup 01] Biskup, J. & Bonatti, P. A. "Lying Versus Refusal for Known Potential Secrets." *Data and Knowledge Engineering 38* (2001): 199-222.

[Butler 02] Butler, S. A. & Fischbeck, P. "Multi-Attribute Risk Assessment." *SREIS 2002, Second Symposium on Requirements Engineering for Information Security*, Raleigh, NC, October 16, 2002, published by CERIAS, Purdue University, Lafayette, IN.

[Clark 93] Clark, R. K.; Greenberg, I. B.; Boucher, P. K.; Lund, T. F.; Neumann, P. G.; Wells, D. M.; & Jenson, E. D. "Effects of Multilevel Security on Real-time Applications," 120-129. *Proceedings of the 9th Annual Computer Security Applications Conference*. Orlando, FL, December 6-10, 1993. Los Alamitos, CA: IEEE Computer Society Press, 1993.

[Davis 93] Davis, Alan. *Software Requirements: Objects, Functions, & States.* Englewood Cliffs, N.J: Prentice-Hall Inc., 1993.

[Ellison 97] Ellison, R. J.; Fisher, D.; Linger, R. C.; Lipson, H. F.; Longstaff, T.; & Mead, N. R. *Survivable Network Systems: An Emerging Discipline* (CMU/SEI-97-TR-013, ADA341963). Pittsburgh, PA: Software Engineering Institute, Carnegie Mellon University, 1997. <http://www.sei.cmu.edu/publications/documents/97.reports/97tr013/97tr013abstract.html>.

[Ellison 03] Ellison, R. J. & Moore, A. P. *Trustworthy Refinement Through Intrusion-Aware Design* (CMU/SEI-2003-TR-002). Pittsburgh, PA: Software Engineering Institute, Carnegie Mellon University, 2003. <http://www.sei.cmu.edu/publications/documents/03.reports/03tr002.html>.

[Feather 03] Feather, M. S. "A Risk-Centric Decision Process." Software Engineering for High Assurance Systems (SEHAS) 2003, Portland, OR, May 9-10, 2003. <http://www.sei.cmu.edu/community/sehas-workshop/feather/>.

[Firesmith 03a] Firesmith, D. G, "Security Use Cases." *Journal of Object Technology 2*, 3 (May-June 2003): 53-64. <http://www.jot.fm/issues/issue_2003_05/column6>.

[Firesmith 03b] Firesmith, D. G "Engineering Security Requirements." *Journal of Object Technology 2*, 1 (January-February 2003): 53-68. <http://www.jot.fm/issues/issue_2003_01/column6>.

[Fu 01] Fu, Z.; Wu, S. F.; Huang, J.; Loh, K.; Gong, F.; Baldine, I.; & Xu, C. "IPSec/VPN Security Policy: Correctness, Conflict Detection and Resolution," 39-56. *Policies for Distributed Systems and Networks*. Edited by M. Sloman, J. Lobo, and E. C. Lupu. Proceedings of the International Workshop, POLICY 2001, Bristol, UK, Jan. 29-31, 2001. Berlin, Germany: Springer-Verlag, 2001 (ISBN 3540416102; Lecture Notes in Computer Science Vol. 1995).

[Gerber 01] Gerber, M.; von Solms, R.; & Overbeek, P. "Formalizing Information Security Requirements." *Information Management and Computer Security 9*, 1 (2001): 32-37.

[Heitmeyer 00] Heitmeyer, C. & Bharadwaj, R. "Applying the SCR Requirements Method to the Light Control Case Study." *Journal of Universal Computer Science 6*, 7 (2000): 650-678.

[Heitmeyer 96] Heitmeyer, C.; Jeffords, R. D.; & Labaw, B.G. "Automated Consistency Checking of Requirements Specifications." *ACM Transactions on Software Engineering and Methodology 5*, 3 (April-June 1996): 231-261.

[Helmer 02] Helmer, G; Wong, J.; Slagell, M.; Honavar, V.; Miller, L.; & Lutz, R. "A Software Fault Tree Approach to Requirements Analysis of an Intrusion Detection System." *Requirements Engineering 7*, 4 (December 2002): 207-220.

[Heninger 78] Heninger, K.; Parnas, D. L.; Shore, J. E.; & Kallander, J. W. "Software Requirements for the A-7E Aircraft." *Technical Report 3876.* Washington, D.C.: Naval Research Laboratory, 1978.

[Heninger 80] Heninger, K. L. "Specifying Software Requirements for Complex Systems: New Techniques and their Application." *IEEE Transactions on Software Engineering SE-6*, 1 (January 1980): 2-13.

[Kienzle 98] Kienzle, D. M. & Wulf, W. A. "A Practical Approach to Security Assessment," 5-16. *Proceedings of the 1998 Workshop on New Security Paradigms*. Charlottesville, VA, Sept. 22-26, 1998. New York, NY: ACM, 1998 (ISBN 0897919866).

[Knorr 01] Knorr, K. & Rohrig, S. "Security Requirements of E-Business Processes," 73-86. *Towards the E-Society: E-Commerce, E-Business, and E-Government*. Edited by B. Schmid, K. Stanoevska-Slabeva, and V. Tschammer. First IFIP Conference on E-Commerce, E-Business, E-Government, Zurich, Switzerland, Oct. 4-5, 2001. Norwell, MA: Kluwer Academic Publishers, 2001 (ISBN 0792375297).

[Labuschagne 00] Labuschagne, L. "A Framework for Electronic Commerce, Security," 441-50. *Information Security for Global Information Infrastructures*. Edited by S. Qing and J. H. P. Eloff. Fifteenth Annual Working Conference on Information Security, Beijing, China, Aug. 22-24, 2000. Norwell, MA: Kluwer Academic Publishers, 2000 (ISBN 0792379144).

[Leitold 99] Leitold, H. & Posch, R. "ATM Network Security: Requirements, Approaches, Standards, and the SCAN Solution," 191-204. *Intelligence in Networks, SMARTNET '99*. Pathumthani, Thailand, Nov. 22-26, 1999. Boston, MA: Kluwer Academic Publishers, 1999.

[Leiwo 99a] Leiwo, J. "A Mechanism for Deriving Specifications of Security Functions in the CC Framework," 416-425. *10th International Workshop on Database and Expert Systems Applications*. Florence, Italy, Sept. 1-3, 1999. Berlin, Germany: Springer-Verlag, 1999.

[Leiwo 99b] Leiwo, J.; Gamage, C.; & Zheng, Y. "Organizational Modeling for Efficient Specification of Information Security Requirements," 247-260. *Advances in Databases and Information Systems: Third East European Conference, ADBIS'99*. Maribor, Slovenia, Sept. 13-16, 1999. Berlin, Germany: Springer-Verlag, 1999 (Lecture Notes in Computer Science Vol. 1691).

[Leveson 95] Leveson, N. G. *Safeware: System Safety and Computers*. Reading, MA: Addison-Wesley, 1995.

[Linger 98] Linger, R. C.; Mead, N. R.; &Lipson, H. F. "Requirements Definition for Survivable Systems," 14-23. *Third International Conference on Requirements Engineering*. Colorado Springs, CO, April 6-10, 1998. Los Alamitos, CA: IEEE Computer Society, 1998.

[Linger 02] Linger, R. C.; Walton, G.; Pleszkoch, M. G.; & Hevner, A. R. "Flow-Service-Quality (FSQ) Requirements Engineering for High Assurance Systems." <http://www.cert.org/archive/pdf/FSQengineeringRHAS-02paper.pdf> (2002).

[Linger 03] Linger, R. C. *Applying Flow-Service-Quality (FSQ) Engineering Foundations to Automated Calculation of Program Behavior* (CMU/SEI-2003-TN-003, ADA412025). Pittsburgh, PA: Software Engineering Institute, Carnegie Mellon University, 2003. <http://www.sei.cmu.edu/publications/documents/03.reports/03tn003.html>.

[McDermott 99] McDermott, J. & Fox, C. "Using Abuse Case Models for Security Requirements Analysis," 55-64. *Proceedings 15th Annual Computer Security Applications Conference*. Scottsdale, AZ, Dec. 6-10, 1999. Los Alamitos, CA: IEEE Computer Society Press, 1999.

[McDermott 01] McDermott, J. "Abuse-Case-Based Assurance Arguments," 366-374. *Proceedings 17th Annual Computer Security Applications Conference*. New Orleans, LA, Dec. 10-14, 2001. Los Alamitos, CA: IEEE Computer Society Press, 2001.

[Mead 01] Mead, N. R.; Linger, R. C.; McHugh, J.; & Lipson, H. F. "Managing Software Development for Survivable Systems." *Annals of Software Engineering 11* (2001): 45-78.

[Mead 02] Mead, N. R. "Survivability Requirements: How Can We Assess Them Versus Other Requirements for High Assurance Systems," 65-68. *International Workshop on Requirements for High Assurance Systems*, Essen, Germany, Sept. 9, 2002. Pittsburgh, PA: Software Engineering Institute, Carnegie Mellon University, 2002.

[Mendiratta 96] Mendiratta, V. B. "Assessing the Reliability Impacts of Software Fault-Tolerance Mechanisms," 99-103. *Proceedings of the 7th International Symposium on Software Reliability Engineering*. White Plains, NY, Oct. 30-Nov. 2, 1996. New York, NY: IEEE Computer Society Press, 1996.

[Mills 92] Mills, H. D. "Certifying the Correctness of Software," 373-381. *Proceedings of the 25th Hawaii International Conference on System Sciences, Vol. 2*. Kauai, Hawaii, Jan. 7-10, 1992. Los Alamitos, CA: IEEE Computer Society Press, 1992.

[Moore 01] Moore, A. P.; Ellison, R. J.; & Linger, R. C. *Attack Modeling for Information Security and Survivability* (CMU/SEI-2001-TN-001, ADA388771). Pittsburgh, PA: Software Engineering Institute, Carnegie Mellon University, 2001. <http://www.sei.cmu.edu /publications/documents/01.reports/01tn001.html>.

[Musa 87] Musa, J. D.; Iannino, A.; & Okumoto, K. *Software Reliability: Measurement, Prediction, and Application.* New York, NY: McGraw-Hill, 1987.

[Ortalo 98] Ortalo, R. "A Flexible Methods for Information System Security Policy Specification," 67-84. *5th European Symposium on Research in Computer Security – Proceedings*. Louvain-la-Neuve, Belgium, Sept. 16-18. Berlin, Germany: Springer-Verlag, 1998. (Lecture Notes in Computer Science Vol. 1485.)

[Paulk 94] Paulk, M. C.; Weber, C. V.; Curtis, B.; & Chrissis, M. B. *The Capability Maturity Model: Guidelines for Improving the Software Process*. Addison-Wesley, Reading, MA, 1994.

[Reiser 00] Reiser, H. & Vogt, G. "Security Requirements for Management Systems using Mobile Agents," 160-165. *Proceedings ISCC2000—Fifth IEEE Symposium on Computers and Communications*. Edited by S. Tohme and M. Ulema. Antibes, France, July 4-7, 2000. Los Alamitos, CA: IEEE Computer Society, 2000 (ISBN 0769507220).

[Sabatier 99] Sabatier, D. & Lartigue, P. "The Use of the B Formal Method for the Design and Validation of the Transaction Mechanism for Smart Card Applications," 348-368. *FM '99: World Congress on Formal Methods, Vol. I*. Toulouse, France, Sept. 20-24, 1999. Berlin, Germany: Springer-Verlag, 1999. (Lecture Notes in Computer Science Vol. 1708.)

[Sawyer 01] Sawyer, P. & Kotonya, G. Ch. 2, "Software Requirements," 9-34. *Guide to the Software Engineering Body of Knowledge, Trial Version 1.00 (SWEBOK)*. Los Alamitos, CA: IEEE Computer Society, 2001. <http://www.swebok.org/>.

[Schneier 00] Schneier, B. *Secrets and Lies: Digital Security in a Networked World*. New York, NY: John Wiley & Sons, 2000.

[SEI 97] Software Engineering Institute. "Requirements Management – Software Capability Maturity Model (SW-CMM) V2.0 Draft." <http://www.sei.cmu.edu/cmm/draft-c/c21rqm.html> (1997).

[SEI 02] Software Engineering Institute. *International Workshop on Requirements for High Assurance Systems*, Essen, Germany, September 9, 2002. Pittsburgh, PA: Software Engineering Institute, Carnegie Mellon University, 2002.

[Sindre 00] Sindre, G. & Opdahl, A. "Eliciting Security Requirements by Misuse Cases," 120-130. *Proceedings of TOOLS Pacific 2000*. Sydney, Australia, Nov. 20-23, 2000. Los Alamitos, CA: IEEE Computer Society Press, 2000.

[Sindre 02] Sindre, G; Opdahl, S.; & Brevik, G. "Generalization/Specialization as a Structuring Mechanism for Misuse Cases", *SREIS 2002, Second Symposium on Requirements Engineering for Information Security*, Raleigh, NC, Oct. 16, 2002, CERIAS, Purdue University, Lafayette, IN.

[Son 98] Son, S. H. & Chaney, C. "Supporting the Requirements for Multi-Level Secure and Real-Time Databases in Distributed Environments," 73-91. *IFIP '98*. Vienna, Austria, Aug. 31-Sept. 4, 1998. Chapman & Hall, 1998.

[SREIS 02] *Second Symposium on Requirements Engineering for Information Security*, Raleigh, NC, October 16, 2002, CERIAS, Purdue University, Lafayette, IN.

[Toval 02] Toval, A.; Nicolas, J.; Moros, B.; & Garcia, F. "Requirements Reuse for Improving Systems Security: A Practitioner's Approach" *Requirements Engineering 6*, 4 (January 2002): 205-219.

[Thayer 97] Thayer, R. & Dorfman, M. *Software Requirements Engineering*, 2nd ed. Los Alamitos, CA: IEEE Computer Society Press, 1997 (ISBN 0-8186-7738-4).

[Trammell 95] Trammell, C. J. "Quantifying the Reliability of Software: Statistical Testing Based on a Usage Model," 208-218. *Proceedings of the Second IEEE International Symposium on Software Engineering Standards*. Montreal, Quebec, Canada, August 21-25, 1995. Los Alamitos, CA: IEEE Computer Society Press, 1995.

REPORT DOCUMENTATION PAGE

Form Approved
OMB No. 0704-0188

1. AGENCY USE ONLY	2. REPORT DATE	3. REPORT TYPE AND DATES COVERED
(Leave Blank)	September 2003	Final

4. TITLE AND SUBTITLE	5. FUNDING NUMBERS
Requirements Engineering for Survivable Systems	F19628-00-C-0003

6. AUTHOR(S)

Nancy R. Mead

7. PERFORMING ORGANIZATION NAME(S) AND ADDRESS(ES)	8. PERFORMING ORGANIZATION REPORT NUMBER
Software Engineering Institute Carnegie Mellon University Pittsburgh, PA 15213	CMU/SEI-2003-TN-013

9. SPONSORING/MONITORING AGENCY NAME(S) AND ADDRESS(ES)	10. SPONSORING/MONITORING AGENCY REPORT NUMBER
HQ ESC/XPK 5 Eglin Street Hanscom AFB, MA 01731-2116	

11. SUPPLEMENTARY NOTES

12A DISTRIBUTION/AVAILABILITY STATEMENT	12B DISTRIBUTION CODE
Unclassified/Unlimited, DTIC, NTIS	

13. ABSTRACT (MAXIMUM 200 WORDS)

This report describes the current state of requirements engineering for survivable systems, that is, systems that are able to complete their mission in a timely manner, even if significant portions are compromised by attack or accident. Requirements engineering is defined and requirements engineering activities are described. Survivability requirements are then explained, and requirements engineering methods that may be suitable for survivable systems are introduced. The report concludes with a summary and a plan for future research opportunities in survivable systems requirements engineering.

14. SUBJECT TERMS	15. NUMBER OF PAGES
requirements engineering, survivable systems, survivability requirements, misuse cases, abuse cases, formal methods, attack trees, fault trees, Software Cost Reduction, requirements reuse, risk analysis	42

16. PRICE CODE

17. SECURITY CLASSIFICATION OF REPORT	18. SECURITY CLASSIFICATION OF THIS PAGE	19. SECURITY CLASSIFICATION OF ABSTRACT	20. LIMITATION OF ABSTRACT
Unclassified	Unclassified	Unclassified	UL